Министерство образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Санкт-Петербургский государственный электротехнический университет “ЛЭТИ” им.В.И.Ульянова (Ленина) »

Кафедра МОЭВМ

Лабораторная работа по ООП №6 по теме “Обработка XML-документов”

Выполнил: Цацкис Артём гр. 3311 Проверил:

Павловский М. Г.

Подпись преподавателя: _____

Санкт-Петербург

2024

Цель работы:

Знакомство с технологией обработки XML-документов и файлов.

Описание задания:

1. Создать проект, который дублирует код лабораторной работы №5
2. Напишите и замените в проекте обработчики кнопок загрузки данных в XML-файл и выгрузки из него.
3. Загрузите данные в экранную форму из XML-файла. Убедитесь, что данные в экранной форме соответствуют данным XML-файла.
4. Внесите изменения в данные экранной формы и сохраните их в XML-файле.
5. Просмотрите в браузере сохраненный XML-файл и убедитесь в правильности работы приложения.

XML-файл до выполнения прогаммы:

```
<?xml version="1.0" encoding="UTF-8"?>
<library>
    <book>
        <title>Война и мир</title>
        <author>Лев Толстой</author>
        <font>Arial</font>
        <pinned>Нет</pinned>
    </book>
    <book>
        <title>1984</title>
        <author>Джордж Оруэлл</author>
        <font>Calibri</font>
        <pinned>Да</pinned>
    </book>
    <book>
        <title>Прощай оружие!</title>
        <author>Эрнест Хемингуэй</author>
        <font>Garamond</font>
        <pinned>Нет</pinned>
    </book>
    <book>
        <title>Убить пересмешника</title>
        <author>Харпер Ли</author>
        <font>Fraktur</font>
        <pinned>Да</pinned>
    </book>
    <book>
        <title>На дороге</title>
        <author>Джек Керуак</author>
        <font>Papyrus</font>
        <pinned>Нет</pinned>
    </book>
</library>
```

XML-файл после выполненя программы:

```
▼<Books>
  ▼<Book>
      <Title>Война и мир</Title>
      <Author>Лев Толстой</Author>
      <Font>Arial</Font>
      <Pinned>Нет</Pinned>
    </Book>
  ▼<Book>
      <Title>1984</Title>
      <Author>Джордж Оруэлл</Author>
      <Font>Calibri</Font>
      <Pinned>Да</Pinned>
    </Book>
  ▼<Book>
      <Title>Прощай оружие!</Title>
      <Author>Эрнест Хемингуэй</Author>
      <Font>Garamond</Font>
      <Pinned>Нет</Pinned>
    </Book>
  ▼<Book>
      <Title>Убить пересмешника</Title>
      <Author>Харпер Ли</Author>
      <Font>Fraktur</Font>
      <Pinned>Да</Pinned>
    </Book>
  ▼<Book>
      <Title>На дороге</Title>
      <Author>Джек Керуак</Author>
      <Font>Papyrus</Font>
      <Pinned>Нет</Pinned>
    </Book>
  ▼<Book>
      <Title>qweqwe</Title>
      <Author>qweqwe</Author>
      <Font>qweqwe</Font>
      <Pinned>Да</Pinned>
    </Book>
  </Books>
```

Код программы:

```
package lab6
```

```
 import java.io.*;
import java.awt.*;
import
java.awt.event.*;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import javax.xml.parsers.DocumentBuilder; import
javax.xml.parsers.DocumentBuilderFactory; import
javax.xml.transform.Transformer; import
javax.xml.transform.TransformerFactory; import
javax.xml.transform.dom.DOMSource; import
javax.xml.transform.stream.StreamResult; import
org.w3c.dom.*;
 public class OOPlab6 {     private
JFrame bookList;     private
DefaultTableModel model;
    private JButton save, add, edit, delete, load;
private JScrollPane scroll;     private JTable
books;

    public void show() {
        bookList = new JFrame("Информация о книгах");
bookList.setSize(600, 400);         bookList.setLocation(100,
100);
        bookList.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        String[] columns = {"Название книги", "Автор", "Шрифт", "Закреплена?"};
        String[][] data = {
            {"Война и мир", "Лев Толстой", "Arial", "Нет"},
            {"1984", "Джордж Оруэлл", "Calibri", "Да"},
            {"Прощай оружие!", "Эрнест Хемингуэй", "Garamond", "Нет"},
            {"Убить пересмешника", "Харпер Ли", "Fraktur", "Да"},
            {"На дороге", "Джек Керуак", "Papyrus", "Нет"}
        };
        model = new DefaultTableModel(data, columns);
books = new JTable(model);
books.setAutoCreateRowSorter(true);         scroll =
new JScrollPane(books);
        bookList.getContentPane().add(scroll, BorderLayout.CENTER);
        JPanel buttonPanel = new JPanel();
save = new JButton("Сохранить");
load = new JButton("Загрузить");         add
= new JButton("Добавить");         edit =
new JButton("Редактировать");         delete
= new JButton("Удалить");
         buttonPanel.add(save);
buttonPanel.add(load);
buttonPanel.add(add);
buttonPanel.add(edit);
buttonPanel.add(delete);
        bookList.getContentPane().add(buttonPanel, BorderLayout.SOUTH);
         save.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
saveToXMLFile();
                JOptionPane.showMessageDialog(bookList, "Данные сохранены.");
```

```
        load.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
                loadFromXMLFile();
                JOptionPane.showMessageDialog(bookList, "Данные загружены.");
            }
        });
         add.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
JTextField titleField = new JTextField();
                JTextField authorField = new JTextField();
                JTextField fontField = new JTextField();
                String[] options = {"Да", "Нет"};
                JComboBox<String> pinnedField = new JComboBox<>(options);
                JPanel panel = new JPanel(new GridLayout(0, 1));
panel.add(new JLabel("Название книги:"));
panel.add(titleField);                  panel.add(new
JLabel("Автор:"));                  panel.add(authorField);
panel.add(new JLabel("Шрифт:"));
panel.add(fontField);
                panel.add(new JLabel("Закреплена?"));
panel.add(pinnedField);

                int result = JOptionPane.showConfirmDialog(bookList, panel, "Добавить
новую книгу",
                        JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE);
if (result == JOptionPane.OK_OPTION) {
                    String title = titleField.getText().trim();
                    String author = authorField.getText().trim();
                    String font = fontField.getText().trim();
                    String pinned = (String) pinnedField.getSelectedItem();
                     if (!title.isEmpty() && !author.isEmpty() && !font.isEmpty())
{                        model.addRow(new Object[]{title, author, font, pinned});
JOptionPane.showMessageDialog(bookList, "Добавлена новая книга.");
                    } else {
                        JOptionPane.showMessageDialog(bookList, "Пожалуйста,
заполните все поля.", "Ошибка", JOptionPane.WARNING_MESSAGE);
                    }
                }
            }
        });

        edit.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
try {
                    int selectedRow = books.getSelectedRow();
                    editBook(selectedRow, "Новое название", "Новый автор", "Новый
шрифт", "Нет");
                    JOptionPane.showMessageDialog(bookList, "Книга
отредактирована.");
                } catch (InvalidBookOperationException ex) {
                    JOptionPane.showMessageDialog(bookList, ex.getMessage(), "Ошибка
редактирования", JOptionPane.ERROR_MESSAGE);
                }
            }
```

```
        });

        delete.addActionListener(new ActionListener() {
```

```
            public void actionPerformed(ActionEvent e) {
try {
                    int selectedRow = books.getSelectedRow();
                    deleteBook(selectedRow);
                    JOptionPane.showMessageDialog(bookList, "Книга удалена.");
                } catch (BookDeletionException ex) {
                    JOptionPane.showMessageDialog(bookList, ex.getMessage(), "Ошибка
удаления", JOptionPane.ERROR_MESSAGE);
                }
            }
        });

        bookList.setVisible(true);
    }
    private void saveToXMLFile() {
        try {
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
Document doc = builder.newDocument();

            Element root = doc.createElement("Books");
doc.appendChild(root);
             for (int i = 0; i < model.getRowCount(); i++)
{                 Element book = doc.createElement("Book");
root.appendChild(book);

                Element title = doc.createElement("Title");
                title.appendChild(doc.createTextNode(model.getValueAt(i,
0).toString()));
                book.appendChild(title);

                Element author = doc.createElement("Author");
                author.appendChild(doc.createTextNode(model.getValueAt(i,
1).toString()));
                book.appendChild(author);

                Element font = doc.createElement("Font");
                font.appendChild(doc.createTextNode(model.getValueAt(i,
2).toString()));
                book.appendChild(font);

                Element pinned = doc.createElement("Pinned");
                pinned.appendChild(doc.createTextNode(model.getValueAt(i,
3).toString()));
                book.appendChild(pinned);
            }
            TransformerFactory transformerFactory = TransformerFactory.newInstance();
            Transformer transformer = transformerFactory.newTransformer();
            DOMSource source = new DOMSource(doc);
            StreamResult result = new StreamResult(new File("books_data.xml"));
             transformer.transform(source,
result);

        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }
    private void loadFromXMLFile() {
```

```
        try {
            File file = new File("books_data.xml");
if (!file.exists()) return;

            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
Document doc = builder.parse(file);

            NodeList nodeList = doc.getElementsByTagName("Book");
model.setRowCount(0); // Очистка текущих данных

            for (int i = 0; i < nodeList.getLength(); i++) {
Node node = nodeList.item(i);
                if (node.getNodeType() == Node.ELEMENT_NODE) {
Element element = (Element) node;

                    String title =
element.getElementsByTagName("Title").item(0).getTextContent();
String author =
element.getElementsByTagName("Author").item(0).getTextContent();
String font =
element.getElementsByTagName("Font").item(0).getTextContent();
String pinned =
element.getElementsByTagName("Pinned").item(0).getTextContent();

                    model.addRow(new Object[]{title, author, font, pinned});
                }
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public void editBook(int rowIndex, String title, String author, String font,
String pinned) throws InvalidBookOperationException {         if (rowIndex < 0
|| rowIndex >= model.getRowCount()) {
            throw new InvalidBookOperationException("Не выбрана книга");
        }
        model.setValueAt(title, rowIndex, 0);
model.setValueAt(author, rowIndex, 1);         model.setValueAt(font,
rowIndex, 2);         model.setValueAt(pinned, rowIndex, 3);
    }      public void deleteBook(int rowIndex) throws
BookDeletionException {         if (rowIndex < 0 || rowIndex >=
model.getRowCount()) {             throw new BookDeletionException("Не
выбрана книга");
        }
        model.removeRow(rowIndex);
    }
    public static void main(String[] args) {
new OOPlab6().show();
    }
}
class InvalidBookOperationException extends Exception {
public InvalidBookOperationException(String message) {
```

```
        super(message);
    }
}    class   BookDeletionException   extends   Exception   {            public
BookDeletionException(String message) {         super(message);
    }
}
```

Ссылка на репозиторий Git-hub:

TsatskisArtem/For-OOP6

OOPlab6.java-код программы

Doc- папка с документацией сгенерированная javadocs

Tsatskis-lab6.pdf-отчет по лабораторной работе

Books_data.xml-файл с книгами